A Scalable and Robust Multi-Agent Approach to Distributed Optimization

Abstract

Modularizing a large optimization problem so that the solutions to the subproblems provide a good overall solution is a challenging problem. In this paper we present a multi-agent approach to this problem based on aligning the agent objectives with the system objectives, obviating the need to impose external mechanisms to achieve collaboration among the agents. This approach naturally addresses scaling and robustness issues by ensuring that the agents do not rely on the reliable operation of other agents.

We test this approach in the difficult distributed optimization problem of imperfect device subset selection [Challet and Johnson, 2002]. In this problem, there are *n* devices, each of which has a "distortion", and the task is to find the subset of those *n* devices that minimizes the average distortion. Our results show that in large systems (1000 agents) the proposed approach provides improvements of over an order of magnitude over both traditional optimization methods and traditional multi-agent methods. Furthermore, the results show that even in extreme cases of agent failures (i.e., half the agents fail midway through the simulation) the system remains coordinated and still outperforms a failure-free and centralized optimization algorithm.

1 Introduction

Modularizing a large optimization problem so that the solutions to the subproblems provide a good overall solution is a challenging problem. Similarly, coordinating a large number of agents to achieve complex tasks collectively presents new challenges to the field of multi-agent systems. In this work we leverage recent advances in the field of multi-agent coordination to modularize and solve a difficult optimization problem.

In truly large problems, many of the reasonable assumptions used in multi-agent coordination problems related to a handful of agents are difficult to justify. When dealing with a small number of agents it is reasonable to assume that agents react to one another, can model one another, and/or enter into contracts with one another [Clement and Durfee, 1999; Decker and Lesser, 1995; Hu and Wellman, 1998; Sandholm and Lesser, 1997]. When dealing with thousands of agents on the other hand, such assumptions become more difficult to maintain. At best each one can assume that the agents are aware of other agents as part of a background. In such cases, agents have to act within an environment that may be shaped by the actions of other agents, but cannot be interpreted as the the by-product of the actions of any single agent.

In this work, we focus on an agent coordination method that aims to handle large systems composed of simple agents and where those agents are failure-prone. The size of the system (e.g., a thousand agents) naturally requires methods that do not rely on detailed information about the actions of all the agents being available to a given agent. The simplicity of the agents requires methods where the interactions among the agents provides the systems' strength rather than the sophistication of each individual agent. Finally, the propensity of the agents to sudden failures requires solutions where one agent's actions are not rigidly dependent on the actions of other agents.

The problem of imperfect device subset selection introduced by Challet and Johnson [Challet and Johnson, 2002] provides the perfect setting to test the efficacy of the multi-agent method. In this problem, there are *n* objects, each of which has a "distortion". The task is to find the subset of those *n* objects that minimizes the average distortion. This is a hard optimization problem, and brute force approaches cannot be used for any but its smallest toy instances [Challet and Johnson, 2002; Garey and Johnson, 1979]. We propose to address this problem by associating each device with a simple adaptive Reinforcement-Learning (RL) agent [Kearns and Koller, 1999; Littman, 1994; Sutton and Barto, 1998]) that decides whether or not its device will be a member of the subset. The essential problem is to determine how best to set the agent utility functions (e.g., subsystem objective functions) in a way that will lead to good values of the global utility (e.g., global objective, in this case average distortion), without involving difficult to scale external mechanisms to ensure cooperation among the agents. This problem has been explored in many different domains, including multi agent systems, distributed optimization, computational economics, mechanism design, computational ecologies and game theory [Boutilier, 1996;

Sandholm and Lesser, 1997; Huberman and Hogg, 1988; Parkes, 2001; Stone and Veloso, 2000]. (See [Tumer and Wolpert, 2004] for a survey of the different approaches taken in each field to this problem.)

This paper presents an agent coordination method well-suited for large and noisy multi-agent systems, and is tested on a difficult distributed optimization problem. In Section 2 we focus on the properties the agent utilities must possess to lead to coordination. In Section 3 we present the imperfect device combination problem and derive the specific agent utilities for this domain. In Section 4 we present results showing that the proposed method outperforms traditional methods by up to an order of magnitude, has superior scaling properties and is resistant to severe cases of agent failure. Finally, in Section 5 we provide a summary and discuss the implications and general applicability of this work.

2 Background

For the joint action of agents working in a large system to provide good values of the global utility, we must ensure that: (i) the agents' goals support the global goal; and (ii) each one has a solvable problem. The first of these two desirable properties is that the agent utilities have to be aligned with the global utility. For discrete states, this can be formalized by assessing the "degree of factoredness" between any two utilities [Agogino and Tumer, 2004], which gives the fraction of actions for an agent where the agent utility and global utility have same delta (e.g., if agent utility goes up so does global utility and vice versa).

The second properties assures that the agent utilities have high "learnability" for the agent. Intuitively, learnability provides the sensitivity of an agent's utility to its own actions. It is computed by dividing the expected value of changes in agent *i*'s utility caused changes in agent *i*'s actions to the expected value of changes in agent *i*'s utility caused by changes in the actions of agents other than *i* [Wolpert and Tumer, 2001; Agogino and Tumer, 2004]. So at a given state, the higher the learnability, the more the utility of agent *i* depends on the move of agent *i*, i.e., the better the associated signal-to-noise ratio for *i*. Higher learnability is desirable because it makes it is easier for *i* to achieve a large values of its utility.

Typically these two requirements are in conflict with one another. As a trivial example, a system in which all the agent utility functions are set to the global utility is fully factored. However, such a system will have low learnability since each agent's utility will depend on the actions of all the other agents in the system. It will be nearly impossible for the agents to determine the best actions to follow in most non-trivial systems. At the other extreme, providing each agent with a simple, local utility function will result in high learnability for the agents' utilities, but will not necessarily lead the system to high values of global utility, unless the degree of factoredness is also high. The challenge we faced then, is to find agent utilities in a given domain, with the best trade-off between these two requirements.

3 Combination of Imperfect Devices

We now present the difficult optimization problem of combining imperfect devices [Challet and Johnson, 2002]. A typical example of this problem arises when many simple and noisy observational devices (e.g., nano or micro devices, low power sensing devices) attempt to accurately determine some value pertinent to the phenomenon they're observing. Each device will provide a single number that is slightly off, similar to sampling a Gaussian centered on the value of the real number. The problem is to choose the subset of a fixed collection of such devices so that the average (over the members of the subset) distortion is as close to zero as possible.

3.1 Problem Definition

Formally, the problem is to minimize

$$\varepsilon \equiv \frac{|\sum_{j=1}^{N} n_j a_j|}{\sum_{k=1}^{N} n_k} \,, \tag{1}$$

where $n_j \in \{0,1\}$ is whether device j is or is not selected, and there are N devices in the collection, having associated distortions $\{a_j\}$. This is a hard optimization problem that is similar to known NP-complete problems such as subset sum or partitioning [Challet and Johnson, 2002; Garey and Johnson, 1979], but has two twists: the presence of the denominator and that $a_j \in R \, \forall j$. In this work we set G the system-level, global utility function to $G = -\varepsilon$ (we do this so that the goal is to "maximize" G, which is more consistent with the concept of "utility" function). G is a function of the full state system z (e.g., joint moves of all the agents).

The system is composed of N agents, each responsible for setting one of the n_j. Each of those agent has its own utility function that it is trying to maximize, though the overall objective is to maximize global performance. Our goal is to devise agent utility functions that will cause the multi-agent system to produce high values of $G(z)$ [Agogino and Tumer, 2004; Wolpert and Tumer, 2001].

3.2 Expected Difference Utility

Now let us present the first of two utilities that possess the desirable properties discussed in Section 2, the **Estimated Difference Utility** (EDU). EDU aims to isolate the impact of an agent on the full system by focusing on the difference between the actual impact the agent has and its "expected" impact. Let, $E_{z_i}[G(z)|z_{-i}]$ provide the expected value of G over the possible actions of agent i, where z_i denotes the state of agent i and z_{-i} denotes the states of all agents other than i. Then EDU for this application becomes:

$$\begin{aligned} EDU_i(z) \quad &\equiv \quad G(z) - E_{z_i}[G(z)|z_{-i}] & (2) \\ &= \quad -\frac{|\sum_{j=1}^{N} n_j a_j|}{\sum_{k=1}^{N} n_k} + \Bigg(p(n_i = 1) \; \frac{|\sum_{j \neq i}^{N} n_j a_j + a_i|}{\sum_{k \neq i}^{N} n_k + 1} \\ &\qquad + p(n_i = 0) \, \frac{|\sum_{j \neq i}^{N} n_j a_j|}{\sum_{k \neq i}^{N} n_k} \Bigg) & (3) \end{aligned}$$

where $p(n_i = 1)$ and $p(n_i = 0)$ give the probabilities that agent i set its n_i to 1 or 0 respectively. In what follows, we will assume that those two actions are equally likely (i.e., for all agents i, $p(n_i = 1) = p(n_i = 0) = 0.5$).

For each agent, EDU is fully factored with G because the second term does not depend on agent i's state [Wolpert and Tumer, 2001] (these utilities are referred to as AU in [Wolpert and Tumer, 2001]). Furthermore, because it removes noise from an agent's utility, EDU yields far better learnability than does G [Wolpert and Tumer, 2001]. This noise reduction is due to the subtraction which (to a first approximation) eliminates the impact of states that are not affected by the actions of agent i.

Depending on which action agent i chose (0 or 1), EDU can be reduced to:

$$EDU_i(z) = 0.5\frac{|\sum_{j=1}^{N} n_j a_j - a_i|}{\sum_{k=1}^{N} n_k - 1} - 0.5\frac{|\sum_{j=1}^{N} n_j a_j|}{\sum_{k=1}^{N} n_k} \quad \text{if } n_i = 1\,,$$

or:

$$EDU_i(z) = 0.5\frac{|\sum_{j=1}^{N} n_j a_j + a_i|}{\sum_{k=1}^{N} n_k + 1} - 0.5\frac{|\sum_{j=1}^{N} n_j a_j|}{\sum_{k=1}^{N} n_k} \quad \text{if } n_i = 0\,.$$

Note that in this formulation, EDU provides a very clear signal. If EDU is positive, the action taken by agent i was beneficial to G, and if EDU is negative, the action was detrimental to G. Thus an agent trying to maximize EDU will efficiently maximize G, without explicitly trying to do so. Furthermore, note that the computation of EDU requires very little information. Any system capable of broadcasting G can be minimally modified to accommodate EDU. For each agent to compute its EDU, the system needs to broadcast the two numbers needed to compute G: the number of devices that were turned on (i.e., the denominator in Equation 1) and the associated subset distortion as a real number (i.e., the numerator in Equation 1 before the absolute value operation is performed. Based on those two numbers, the agent can compute its EDU.

3.3 Wonderful Life Utility

The second utility we present is the **Wonderful Life Utility** (WLU) which aims to isolate the impact of an agent on the full system by focusing on the difference between the impact of the agent and its "disappearance" from the system [Wolpert and Tumer, 2001]. WLU for this application becomes:

$$\begin{aligned} WLU_i(z) &\equiv G(z) - G(z_{-i}) \\ &= -\frac{|\sum_{j=1}^{N} n_j a_j|}{\sum_{k=1}^{N} n_k} + \frac{|\sum_{j\neq i}^{N} n_j a_j|}{\sum_{k\neq i}^{N} n_k} \end{aligned} \tag{4}$$

The major difference between EDU and WLU is in how they handle the noise removing second term. EDU provides an estimate of agent i's impact by sampling all possible actions of agent i whereas WLU simply removes agent i from the system. WLU is also factored with G, because the second term does not depend on the actions of agent i (ie., both WLU and G have the same derivative with respect to z_i, the state of agent i [Wolpert and Tumer, 2001]. Note however, that unlike with EDU, the action chosen by agent i has a large impact on the efficiency of WLU. If agent i chooses action 0, the two terms in Equation 4 are identical, resulting in a WLU of zero. Depending on which action agent i chose (0 or 1), WLU can be reduced to:

$$WLU_i(z) = \frac{|\sum_{j=1}^{N} n_j a_j - a_i|}{\sum_{k=1}^{N} n_k - 1} - \frac{|\sum_{j=1}^{N} n_j a_j|}{\sum_{k=1}^{N} n_k} \quad \text{if } n_i = 1\,, \tag{5}$$

or:

$$WLU_i(z) = 0 \quad \text{if } n_i = 0\,. \tag{6}$$

In this formulation, unlike EDU, WLU provides a clear signal only if agent i had chosen action 1. In that case, a positive WLU means that the action was beneficial to G, and a negative WLU means that the action was detrimental for G. However, if agent i had chosen action 0, it receives a reward of 0 regardless of whether that action was good or bad for G. This means that on average half the actions an agent takes will be random as far as G is concerned. Considering learnability implications, this means that on average WLU will have half the learnability of EDU for this problem.

4 Experimental Results

In this work we purposefully used computationally unsophisticated and easy to build agents for the following reasons:

1. To ensure that we remained consistent with our purpose of showing that a large scale system of potentially failure-prone agents can be coordinated to achieve a global goal. Indeed, building thousands of sophisticated agents may be prohibitively difficult; therefore though systems that will scale up to thousands may use sophisticated agents, they cannot rely on such sophistication.
2. To focus on the design of the utility functions. Having sophisticated agents can obscure the differences in performance due to the agent utility functions and the algorithms they ran. By having each agent run a very simple algorithm we kept the emphasis on the effectiveness of the utility functions.

Each agent had a data set and a simple reinforcement learning algorithm. Each agents' data set contained $\{time, action, utility\ value\}$ triplets that the agent stored throughout the simulation. At each time step each agent chose what action to take, which provided a joint action which in turn set the system state. Based on that state the global utility, and the agent utility for all agents were computed. The new $\{time, action, utility\ value\}$ for agent i is then added to the data set maintained by agent i. This is done for all agents and then the process repeats.

To choose its actions, an agent uses its data set to estimate the values of the utility it would receive for taking each of its two possible move. Each agent i picks its action at a time step based on the utility estimates it has for each possible action. Instead of simply picking the largest estimate, to promote exploration it probabilistically selects an action, with a higher likelihood of selecting the actions with higher utility estimates, e.g., it uses a Boltzmann distribution across the utility values [Sutton and Barto, 1998]. Because the experiments were run for short periods of time, the temperature in the Boltzmann distribution did not decay in time. Finally, to form the agents' initial data sets, there is an initialization period in which all actions by all agents are chosen uniformly randomly, with no learning used. It is after this initialization period ends that the agents choose their actions according to the associated Boltzmann distributions.

For all learning algorithms, the first 20 time steps constitute the data set initialization period (note that all learning algorithms must "perform" the same during that period, since none are actually in use then). Starting at $t = 20$, with each consecutive time step a fixed fraction of the agents switch to using their learner algorithms instead, while others continue to take random actions. Because the behavior of the agents starting to use their learning algorithm changes, having all agents start learning simultaneously provides a sudden "spike" into the system which significantly slows down the learning process. This gradual introduction of the learning algorithms is intended to soften the "discontinuity" in each agent's environment. In these experiments, for $N = 50$ and $N = 100$, three agents turned on their learning algorithms at each time step, and for $N = 1000$, sixty agents turned on their learning algorithms at each time step.

4.1 Agent Utility Performance

Figures 1-2 show the convergence properties of different agent utilities and a search algorithm in systems with 100 and 1000 agents respectively. The results reported are based on 20 different $\{a_i\}$ configurations, where each $\{a_i\}$ is selected from a Gaussian distribution with zero mean and unit variance. For each configuration, the experiments were run 50 times (i.e., each point in the graphs is the average of $20 \times 50 = 1000$ runs). In all cases, the Boltzmann parameter (e.g., temperature τ) was set to 0.1. The graphs labeled G, EDU and WLU show the performance of agents using reinforcement learners with those reinforcement signals provided by G, EDU and WLU respectively. S shows the performance of local search where new n_i's are generated at each step by perturbing the current state and selected if the solution is better than the current best solution (in the experiments reported here, 25% of the actions were randomly changed at each time step, though somewhat surprisingly, the results are not particularly sensitive to this parameter). Because the runs are only 200 time steps long, algorithms such as simulated annealing do not outperform local search: there is simply no time for an annealing schedule. This local search algorithm provides the performance of an algorithm with centralized control.

In both cases in which agents use the G utility, they have a difficult time learning. The noise in the system is too large for such agents to learn how to select their actions. For 100 agents (Figure 1), WLU performs at the level of the centralized algorithm. Because agents only receive useful feedback when they take one of the two actions, the noise in the system is larger than that for EDU. This "noise" becomes too much for systems with 1000 agents (Figure 2), where WLU is outperformed by the centralized algorithm. EDU, on the other hand, continues to provide a clean signal for all systems up to the largest we tested (1000 agents). Note that because agents turning on their learning algorithm changes the environment, the performance of the system as whole degrades immediately after learning starts (i.e., after 20 steps) in some cases. Once agents adjust to the new environment, the system settles down and starts to converge.

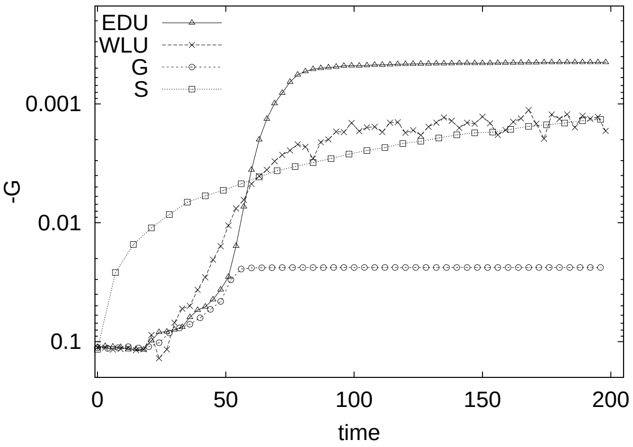

Figure 1: Performance of the three utility functions for N=100.

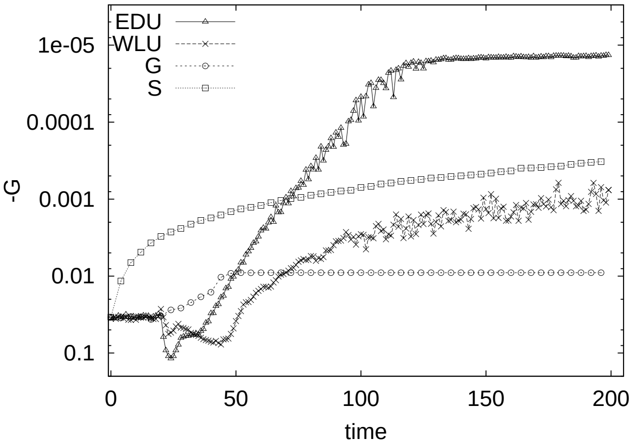

Figure 2: Performance of the three utility functions for N=1000.

4.2 Scaling Characteristics of Utilities

Figure 3 shows scaling results (the $t = 200$ average performance over 1000 runs) along with the associated error bars (differences in the mean). As N grows two competing factors come into play. On the one hand, there are more degrees of freedom to use to minimize G. On the other hand, the problem becomes more difficult: the search space gets larger for S, and there is more noise in the system for the learning algorithms. To account for these effects and calibrate the performance values as N varies, we also provide the baseline performance of the "algorithm" that randomly selects its action ("Ran"). Note that the difference between the performances of all algorithms and EDU increases when the system size increases, reaching a factor of twenty for S and over 600 for G for $N = 1000$.

Also note that all algorithms but EDU have slopes similar to that of "Ran", showing that they cannot use the additional degrees of freedom provided by the larger N. Only

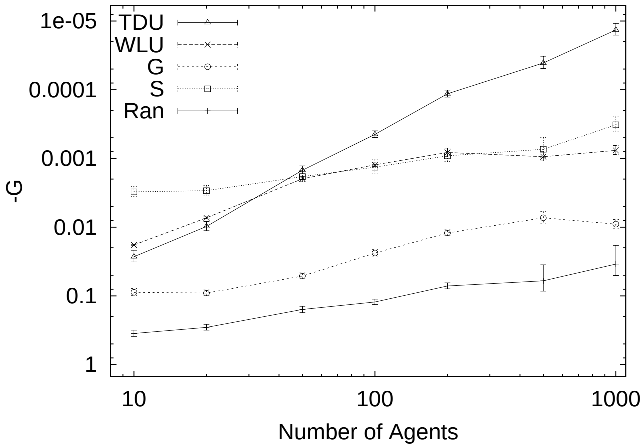

Figure 3: Scaling performance of the three utility functions.

EDU effectively uses the new degrees of freedom, providing gains that are proportionally higher than the other algorithms (i.e., the rate at which *EDU*'s performance improves outpaces what is "expected" based on the *Ran*'s performance).

4.3 Robustness

In order to evaluate the robustness of the proposed utility functions for multiagent coordination, we tested the performance of the system when a subset of the agents failed during the simulation. At a given time ($t = 100$ in these experiments), a certain percentage of agents failed (e.g., were turned off) simulating hazardous condition in which the functioning of the agents cannot be ascertained. The relevance of this experiment is in determining whether the proposed utility functions require all or a large portion of the agents to perform well to be effective, or whether they can handle sudden changes to their environment.

Figure 4 shows the performance of EDU, WLU, and G for 100 agents when 20% of the agents fail at time step $t = 100$. The results of the centralized search algorithm with no failures ("S" from Section 4.1), is also included for comparison. In these experiments, none of the agent learning algorithms were adjusted to account for the change in the environment. In agents that continued to function, the learning proceeded as though nothing had happened. As a consequence, not only did the agents need to overcome the sudden change in their task but they had to do so with parameters tuned to the previous environment. Despite these limitations, EDU recovers rapidly for the 100 agent case, whereas G and WLU do not. Note this is a powerful results: a distributed algorithm with only 80% functioning agents tuned to a different environment outperforms a 100% functioning centralized algorithm.

Figure 5 show the performance of EDU when the percentage of agent failures increases from 10 to 50% for 100 agents. For comparison purposes, the search results (from Section 4.1) are also included. After the initial drop in performance when the agents stop responding, EDU trained algorithms recover rapidly and even with half the agents outperform the fully functioning and centralized search algorithm.

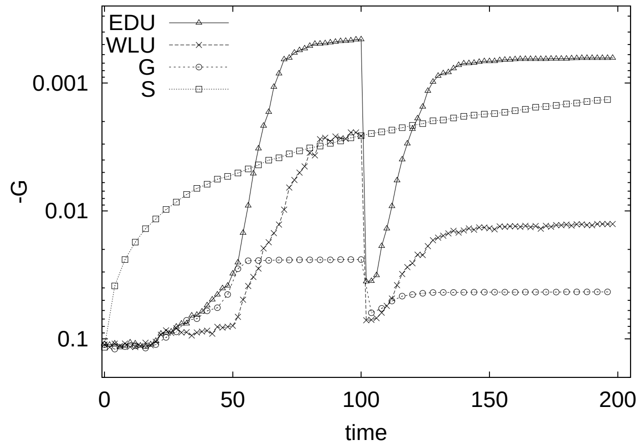

Figure 4: Performance of the three utilities for 100 agents, 20% of which fail at time t=100 (S has no failures).

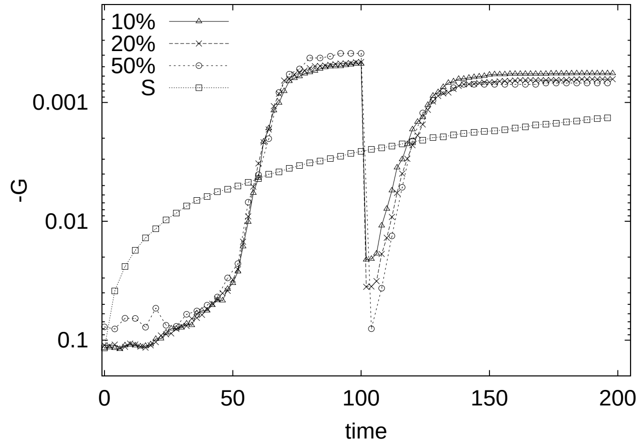

Figure 5: Effect of agent failures on EDU for 100 agents (S has no failures).

These results demonstrate both the adaptability of the EDU and its robustness to failures of individual agents, even in extreme cases.

5 Discussion

The combination of imperfect devices is a simple abstraction of a problem that will loom large in the near future: How to coordinate a very large numbers of agents – with limited sophistication and failure prone – to achieve a prespecified global objective. This problem is fundamentally different from traditional multi-agent coordination (and distributed optimization) problems in at least three ways: (i) the agents are simple and do not model the actions of other agents; (ii) the agents are unreliable and failure-prone; and (iii) the number of agents is in the thousands.

The work summarized in this paper is based on ensuring coordination while eliminating external mechanisms such as contracts and incentives to allow the systems to scale. In the experimental domain of selecting a subset of imperfect devices, the results show the promise of this method by providing performance improvements of twenty fold over a centralized algorithm and of nearly three orders of magnitude over a multi-agent system using the global utility (G) directly. Furthermore, when as many as half the agents fail during simulations, the proposed method still outperforms a fully functioning centralized search algorithm.

This approach is well-suited for addressing coordination in large scale cooperative multi-agent systems where the agents do not have pre-set and possibly conflicting goals, or when the agents do not need to hide their objectives. The focus is on ensuring that the agents do not inadvertently frustrating one another in achieving their goals. The results show that in such large scale, failure-prone systems, this method performs well precisely because it does not rely on the agents building an accurate model of their surroundings, modeling the actions of other agents or requiring all agents in the system to reach a minimum performance level.

References

[Agogino and Tumer, 2004] A. Agogino and K. Tumer. Unifying temporal and structural credit assignment problems. In *Proc. of the Third Intl Jt. Conf. on Autonomous Agents and Multi-Agent Systems*, New York, NY, July 2004.

[Arai *et al.*, 2000] S. Arai, K. Sycara, and T. Payne. Multi-agent reinforcement learning for planning and scheduling multiple goals. In *Proc. of the Fourth Intl Conference on MultiAgent Systems*, pages 359–360, July 2000.

[Boutilier, 1996] C. Boutilier. Planning, learning and coordination in multiagent decision processes. In *Proceedings of the Sixth Conference on Theoretical Aspects of Rationality and Knowledge*, Holland, 1996.

[Challet and Johnson, 2002] D. Challet and N. F. Johnson. Optimal combinations of imperfect objects. *Physical Review Letters*, 89:028701, 2002.

[Clement and Durfee, 1999] B. Clement and E. Durfee. Theory for coordinating concurrent hierarchical planning agents. In *Proceedings of the National Conference on Artificial Intelligence*, pages 495–502, 1999.

[Decker and Lesser, 1995] K. Decker and V. Lesser. Designing a family of coordination mechanisms. In *Proceedings of the International Conference on Multi-Agent Systems*, pages 73–80, June 1995.

[Fredslund and Mataric, 2002] J. Fredslund and M. J. Mataric. Robots in formation using local information. In *Proc., 7th Intl Conf. on Intelligent Autonomous Systems*, pages 100–107, Marina del Rey, CA, March 2002.

[Garey and Johnson, 1979] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, San Fransisco, 1979.

[Hogg and Huberman, 1998] T. Hogg and B. A. Huberman. Controlling smart matter. *Smart Materials and Structures*, 7:R1–R14, 1998.

[Hu and Wellman, 1998] J. Hu and M. P. Wellman. Multi-agent reinforcement learning: Theoretical framework and an algorithm. In *Proc. of the Fifteenth Intl Conference on Machine Learning*, pages 242–250, June 1998.

[Huberman and Hogg, 1988] B. A. Huberman and T. Hogg. The behavior of computational ecologies. In *The Ecology of Computation*, pages 77–115. North-Holland, 1988.

[Kearns and Koller, 1999] M. Kearns and D. Koller. Efficient reinforcement learning in factored MDPs. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, pages 740–747, 1999.

[Kraus, 1997] S. Kraus. Negotiation and cooperation in multi-agent environments. *Artificial Intelligence*, pages 79–97, 1997.

[Littman, 1994] M. L. Littman. Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of the 11th International Conference on Machine Learning*, pages 157–163, 1994.

[Parkes, 2001] D. C. Parkes. *Iterative Combinatorial Auctions: Theory and Practice*. PhD thesis, University of Pennsylvania, 2001.

[Pynadath and Tambe, 2002] D. Pynadath and M. Tambe. The communicative multiagent team decision problem: Analyzing teamwork theories and models. *Journal of Artificial Intelligence Research*, 16:389–423, 2002.

[Sandholm and Lesser, 1997] T. Sandholm and V. R. Lesser. Coalitions among computationally bounded agents. *Artificial Intelligence*, 94:99–137, 1997.

[Scerri *et al.*, 2004] P. Scerri, Y. Xu, E. Liao, J. Lai, and K. Sycara. Scaling teamwork to very large teams. In *Proc. of the Third Intl Joint Conference on Autonomous Agents and Multi-Agent Systems*, New York, NY, July 2004.

[Sen *et al.*, 1994] Sandip Sen, Mahendra Sekaran, and John Hale. Learning to coordinate without sharing information. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pages 426–431, Seattle, WA, 1994.

[Stone and Veloso, 2000] P. Stone and M. Veloso. Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots*, 8(3), 2000.

[Stone, 2000] P. Stone. *Layered Learning in Multi-Agent Systems: A Winning Approach to Robotic Soccer*. MIT Press, Cambridge, MA, 2000.

[Sutton and Barto, 1998] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.

[Tambe, 1997] M. Tambe. Towards flexible teamwork. *Journal of Artificial Intelligence Research*, 7:83–124, 1997.

[Tumer and Wolpert, 2004] K. Tumer and D. Wolpert, editors. *Collectives and the Design of Complex Systems*. Springer, New York, 2004.

[Wolpert and Tumer, 2001] D. H. Wolpert and K. Tumer. Optimal payoff functions for members of collectives. *Advances in Complex Systems*, 4(2/3):265–279, 2001.